PROGRAMANDO MICROCONTROLADORES PIC EN LENGUAJE C

PIC Microcontrollers Programming in C language

CARLOS ALBERTO HENAO
Tecnólogo Eléctrico
Estudiante de Ingeniería Eléctrica
caramelo@utp.edu.co

EDISON DUQUE
Ingeniero Electrónico, M.Sc
Profesor Asociado
Universidad Tecnológica de Pereira
eduque@utp.edu.co

RESUMEN
En este artículo se presenta el manejo del compilador PCW para microcontroladores PIC, el cual permite programar fácilmente los microcontroladores en lenguaje C, con lo cual se realiza más rápidamente el desarrollo de programas que serían bastante complejos de abordar en lenguaje ensamblador.

PALABRAS CLAVES: Compilador, lenguaje C, microcontrolador.

ABSTRACT
In this article management PCW compiler for PIC microcontrollers, which can easily program the PIC microcontrollers in C language, which is quickly developing programs that would be quite complex to make in assembly language.

KEYWORDS: *Compiler, C language, microcontroller.*

1. INTRODUCCIÓN

Tradicionalmente muchos programadores de microcontroladores PIC utilizan el lenguaje ensamblador para realizar sus proyectos, pero en la actualidad existen compiladores de lenguajes de alto nivel que permiten realizar las mismas tareas en un menor tiempo de desarrollo y con mucha mayor facilidad en la programación.

El *PCW Compiler* es una herramienta útil para programar microcontroladores PIC, en la cual están incluidas las librerías para manejar una pantalla LCD, el protocolo de comunicación serial, manejo de puertos, etc. En la actualidad el compilador PCW es una herramienta en desarrollo, poco a poco se han ido adaptando nuevas librerías, nuevos microcontroladores y nuevas ayudas.

2. CARACTERÍSTICAS DE COMPILADOR PCW

- Traduce el código C del archivo fuente (.c) a lenguaje de máquina para programar microcontroladores PIC (.HEX).
- Se incluye Drivers o librerías de código fuente para manejo de pantallas LCD, teclados, sensores, protocolos de comunicación, memorias, conversión analógico a digital, etc.
- Se integra al módulo de desarrollo IDE del MPLAB (software de desarrollo de Microchip) y otros simuladores y editores para la depuración del código fuente.
- Funciones para el manejo de interrupciones.

Fecha de Recepción: 15 de Septiembre de 2009.
Fecha de Aceptación: 12 de Octubre de 2009

2.1 EL COMPILADOR PCW

Básicamente el compilador PCW maneja la misma estructura de programación que el lenguaje de programación C, las similitudes más importantes son:

2.1.1 OPERADORES ARITMÉTICOS

Los operadores aritméticos del PCW son praticamenete los mismos que su homólogo el lenguaje C y son los siguientes:

Operador	Descripción	Ejemplo
+	Suma (enteros)	Suma = a + b
-	Resta (enteros)	Resta = a - b
*	Producto (enteros)	Produ= a * b
/	División (enteros)	div = a / b
%	Módulo: (residuo)	Mod = a % b

Tabla 2.1 Operadores aritméticos

2.1.2 OPERADORES RELACIONES

Los operadores relacionales son los siguientes

Operador	Descripción
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual que

Tabla 2.2: Operadores relaciones

2.1.3 FORMATOS DE LAS VARIABLES

C	Caracter
U	Entero sin signo
x	Entero en Hex (en minúscula)
X	Entero en Hex (en mayúscula)
D	Entero sin signo
%e	Real en formato exponencial
%f	Real (float)
Lx	Entero largo en Hex (en minúscula)
LX	Entero largo en Hex (en mayúscula)
Lu	Decimal largo sin signo
Ld	Decimal largo con signo

Tabla 2.3: Formatos de las variables

2.1.4 SENTENCIAS BÁSICAS Y BUCLES

El compilador PCW contiene los bucles y sentencias básicas del lenguaje C que son principalmente las siguientes:

- Sentencia if (expresión)
- Sentencia if..... Else
- Bucle while
- Bucle for
- Bucle do... while

2.2 COMPILADOR PCB Y PCM PCH

Esta herramienta contiene 3 compiladores que son:

PCB: Genera código para microcontroladores de 12 bits en memoria de programa (ejemplo: la familia PIC12C54x).

PCM: Genera código para microcontroladores de 14 bits en memoria de programa (ejemplo: la familia PIC16F87x).

PCH: Genera código para microcontroladores de 16 bits en memoria de programa.

En la siguiente gráfica se muestra la función de los compiladores

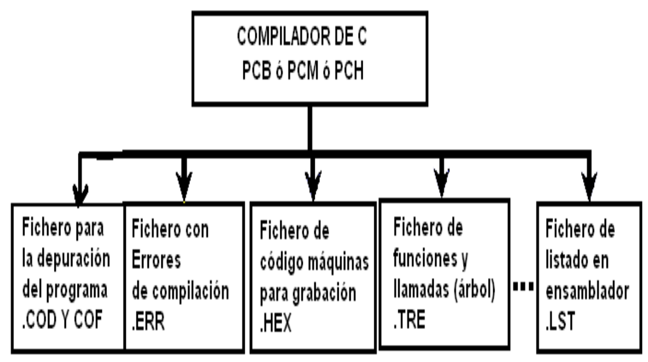


Figura 2.1 : Función de los compiladores

2.3 DIRECTIVAS EN EL CCS

#INCLUDE <NOMBRE_DEL_FICHERO>

Esta directiva hace que el compilador incluya en el fichero fuente el texto que contiene el archivo indicado.

Ejemplo: #include <16F877.H>

#FUSE

Esta directiva define qué fusibles deben activarse en el dispositivo cuando se programe. Esta directiva no afecta a la compilación; sin embargo, esta información se pone en el archivo de salida. Algunas de las opciones más usadas son:

LP, XT, HS, RC (Tipo de oscilador)
WDT, NOWDT (Activación del Watch Dog Timer)
PROTECT, NOPROTECT (Protección del código)
PUT, NOPUT (Temporizador de arranque)
BROWNOUT, NOBROWNOUT (Detección de caídas de tensión de la fuente de alimentación)

Ejemplo #fuse HS, WDT.

#INT_XX

Estas directivas especifican que la función que le sigue es una función de interrupción. Las funciones de interrupción no pueden tener ningún parámetro. Como es natural, no todas las directivas pueden usarse con todos los dispositivos. Las directivas más comunes son las siguientes:

#INT_EXT: Interrupción externa

#INT_TRCC: Desbordamiento del TIMER0 (RTCC)

#INT_RB: Cambio en los pines B4, B5, B6, B7

#INT_AD: Conversor A/D

#INT_TIMER1: Desbordamiento del TIMER1.

#INT_TIMER2: Desbordamiento del TIMER2

#INT_CP1: Modo captura de datos por CCP1

#INT_CCP2: Modo captura por CCP2

#USE DELAY (Clock = Frecuencia):

Esta directiva indica al compilador la frecuencia del procesador, en ciclos por segundo, a la vez que habilita el uso de las funciones DELAY_MS() y DELAY_US().

Ejemplo: #USE DELAY (CLOCK = 4000000)

#USE STANDARD_io (Puerto)

Esta directiva afecta al código que el compilador generará para las instrucciones de entrada y salida. Este método rápido de hacer I/O ocasiona que el compilador realice I/O sin programar el registro de dirección. El puerto puede ser A-G.

Ejemplo: #USE STANDARD_io(B)

#USE RS232 (BAUD = baudios, XMIT = pin, RCV= pin)

Esta directiva le dice al compilador la velocidad en bits por segundo y los pines utilizados para la comunicación serie. Esta directiva tiene efecto hasta que se encuentra otra directiva RS232.

La directiva **#USE DELAY** debe aparecer antes de utilizar **#USE RS232**. Esta directiva habilita el uso de funciones tales como GETCH, PUTCHAR y PRINTF.

2.4 FUNCIONES DISCRETAS DE I/0

Input(pin)

Devuelve el estado '0' o '1' de la patilla indicada en pin. El método de acceso de I/O depende de la última directiva #USE *_IO utilizada. El valor de retorno es un entero corto.

Ejemplo : if (Input(Pin_B0)==1)

Output (Pin, Value)

Esta función saca el bit dado en value(0 o 1) por la patilla de I/O especificada en pin. El modo de establecer la dirección del registro, está determinada por la última directiva #USE *_IO.

Ejemplo : output_bit(PIN_B0, 0);

Output _high(pin)

Pone a 'uno' el pin indicado. El método de acceso de I/O depende de la última directiva #USE *_IO utilizada.

Ejemplo : Output_high(PIN_C0)

Output_low(pin)

Pone a 'cero' el pin indicado. El método de acceso de I/O depende de la última directiva #USE *_IO.

Ejemplo : Output_low(PIN_D0)

Port_b_pullups(flag*)*

Esta función activa/desactiva las resistencias pullups en las entradas del puerto B. Flag puede ser TRUE (activa) o FALSE (desactiva).

Ejemplo: Port_b_pullups(false)

Set_ tris_puerto(Valor)

Estas funciones permiten escribir directamente los registros tri-estado para la configuración de los puertos (configurar pines de entrada y salida).

Esto debe usarse con FAST_IO(), cuando se accede a los puertos de I/O como si fueran memoria, igual que cuando se utiliza una directiva #BYTE. Cada bit de value representa una patilla. Un '1' indica que la patilla es de entrada y un '0' que es de salida.

Ejemplo : Set_tris_A(0xff); puerto A como entrada

2.5 FUNCIONES DE RETARDO

Delay_cicles(Valor)

Esta función realiza retardos según el número de ciclos de instrucción especificado en count; los valores posibles van desde 1 a 255. Un ciclo de instrucción es igual a cuatro periodos de reloj.

Ejemplo : Delay_cicles(100); Cuenta 100 ciclos

Delay_ms(Valor)

Esta función realiza retardos del valor especificado en time. Dicho valor de tiempo es en milisegundos y el rango es 0-65535.

Para obtener retardos más largos así como retardos 'variables' es preciso hacer llamadas a una función separada; véase el ejemplo siguiente.

Delay_us(Valor)

Esta función realiza retardos del valor especificado en time. Dicho valor es en microsegundos y el rango va desde 0 a 65535. Es necesario utilizar la directiva #use delay antes de la llamada a esta función para que el compilador sepa la frecuencia de reloj.

2.6 FUNCIONES PARA LA MANIPULACIÓN DE BITS

Bit_clear (Var, Bit)

Esta función simplemente borra (pone a '0') el dígito especificado en bit(0-7 ó 0-15) del byte o palabra aportado en var. El bit menos significativo es el 0.

Ejemplo : int x = 10;

Bit_clear(x,0) ;

Bit_set(Var, bit)

Esta función pone a '1' el dígito especificado en bit(0-7 o 0-15) del byte o palabra aportado en var.

Rotate_left(Dirección, bytes)

Esta función rota a la izquierda un bit de un array o de una estructura. Nótese que la rotación implica que el bit MSB pasa a ser el bit LSB. Dirección puede ser un identificador de un array o la dirección a un byte o a una estructura, por ejemplo, &dato. bytes es el número de bytes implicados en la rotación.

Ejemplo : X = 0*50
Rotate_left(&X, 1) ; 0*A0

Rotate_right (Dirección, bytes)

Esta función rota a la derecha un bit de un array o de una estructura. Nótese que esta rotación implica que el bit LSB pasa a ser el bit MSB. address puede ser un identificador de un array o la dirección a un byte o a una estructura, por ejemplo, &dato. bytes es el número de bytes implicados en la rotación.

2.7 MANEJO DEL PROTOCOLO RS232

GETC() , GETCH(), GETCHAR()

Estas funciones esperan un carácter por la patilla RCV del dispositivo RS232 y retorna el caracter recibido.

Es preciso utilizar la directiva #USE RS232 antes de la llamada a esta función para que el compilador pueda determinar la velocidad de transmisión y la patilla utilizada. La directiva #USE RS232 permanece efectiva hasta que se encuentre otra que anule la anterior.

Los procedimientos de I/O serie exigen incluir #USE DELAY para ayudar a sincronizar de forma correcta la velocidad de transmisión. Se debe tener en cuenta que es necesario adaptar los niveles de voltaje antes de conectar el PIC a un dispositivo RS-232.

PUT() , PUTCHAR()

Estas funciones envían un caracter a la patilla XMIT del dispositivo RS232. Es preciso utilizar la directiva #USE RS232 antes de la llamada a esta función para que el compilador pueda determinar la velocidad de transmisión y la patilla utilizada. La directiva #USE RS232 permanece efectiva hasta que se encuentre otra que anule la anterior.

Printf ([funtion], string, [valor])

La función de impresión formateada PRINTF saca una cadena de caracteres al estándar serie RS-232 o a una función especificada.

Cuando se usan variables, string debe ser una constante. El carácter % se pone dentro de string para indicar un valor variable, seguido de uno o más caracteres que dan formato al tipo de información a representar.

2.8 MANEJO DEL MODULO LCD

#INCLUDE <LCD.C>

Librería que maneja el módulo LCD, se debe poner siempre que se vaya a utilizar una de estas pantallas o displays. Por defecto se utiliza conexión a 4 bitas entre el microcontrolador y la pantalla LCD.

Al puerto asignado para el manejo del módulo LCD se le asigna por defecto la siguiente configuración de conexión hacia la LCD.

Puert_X0 = Al terminal E de la LCD
Puerto_X1 = Al terminal RW de la LCD
Puerto_X2 = Al terminal RS de la LCD
Puerto_X4-X7 = Al los terminales D4 –D7 respectivamente
Puerto_X3 = No se conecta

Donde X puede ser el puerto (A, B, C, D, E ,G)

PRINTF(LCD_PUTC,"[STRING]")

Está función permite escribir en la LCD, es decir, poner caracteres que se vayan a visualizar en el modulo LCD.

Ejemplo : Printf(lcd_putc,"Hola Mundo"); escribe Hola Mundo en la LCD

LCD_INIT()

Esta función inicializa la LCD (borra toda la pantalla)

Ejemplo : do

```
{
    lcd_init();
    printf(lcd_putc,"Microcontroladores");
    delay_ms(1000);
}
while(true);
```

3.0 EJEMPLOS DE PROGRAMACIÓN

3.1 EJEMPLO MODULO LCD

En el presente ejemplo se muestra como escribir en una pantalla LCD utilizando el compilador PCW, los pasos son los siguientes:

- Definir que tipo de compilador (PCB, PCM, PCH) y que tipo de microcontrolador se va a utilizar.
- Definir la frecuencia del oscilador
- Definir los puertos por donde se van a conectar la LCD
- Inicializar la LCD
- Escribir en la LCD
- Esperar un tiempo
- Volver a empezar el ciclo

Las conexiones se muestran en la figura 3.1

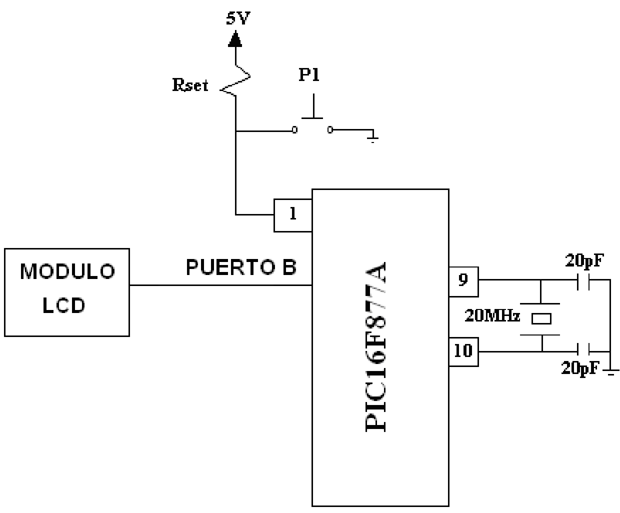


Figura 3.1: Diagrama de conexiones de modulo LCD

El código es siguiente:

```
#if defined(__PCM__)
#include <16f877.h>
#use delay(clock = 4000000)
#define use_portb_lcd TRUE
#include <lcd.c>

void main()
{
  do
   {
     lcd_init();          // se inicializa la lcd
     lcd_gotoxy(5,1);  // ubica puntero
     printf(lcd_putc,"MICROCHIP");
     delay_ms(1000);
   }
  while(true);
}
```

3.2 EJEMPLO MANEJO DEL PROTOCOLO RS232

En el siguiente ejemplo dan las pautas para programar el microcontrolador para que envie datos utilizando el protocolo RS-232 con el compilador PCW, los pasos son los siguientes:

- Definir que tipo de compilador (PCB, PCM, PCH) y que tipo de microcontrolador se a utilizar.
- Definir la frecuencia del oscilador
- Configurar el protocolo RS-232 utilizando la directiva #USE RS232 (BAUD = bits por segundo, XMIT = pin, RCV= pin).
- Mandar dato utilizando la instrucción printf([Funtion], string, [valor])

En el ejemplo se utilizó un PIC16F877 que contiene en su arquitectura un USART (módulo de comunicaciones seriales) para la comunicación con otros dispositivos. En la figura 3.2 se muestran la conexiones.

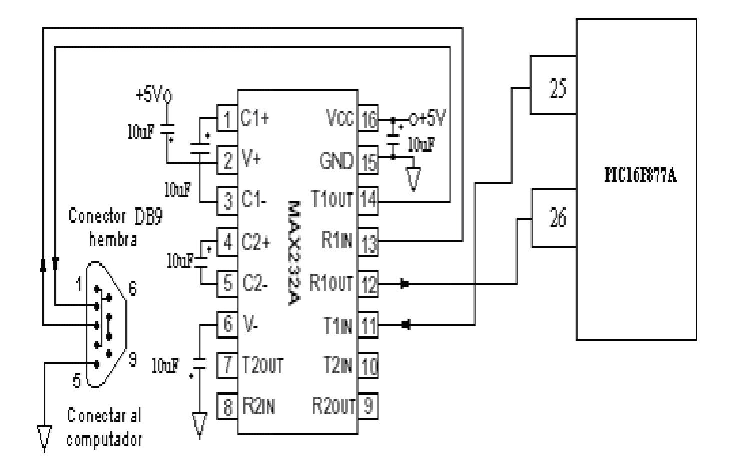


Figura 3.2. Comunicación serial RS-232.

Se transmite por el Puerto C bit 6 y se recibe por el Puerto C pin 7.

El programa es el siguiente:

```
#if defined(__PCM__)
#include <16f877.h>
#use delay(clock = 4000000)
#use rs232(baud = 2400, xmit = PIN_C6, rcv = PIN_C7)
#use standard_io(c)

void main()
 {
   do
    {
      printf("\n MICROCONTROLADORES PIC");
    }
   while(true);
    }
```

Este pequeño programa tiene como función enviar la cadena de caracteres “MICROCONTROLADORES PIC” hacia otro dispositivo, por ejemplo otro microcontrolador, un controlador lógico programable o una computadora con puerto serial. Si lo que se quiere es enviar algún valor numérico se deben utilizar los formatos de la tabla 2.3. Los ejemplos antes mencionados se pueden simular en el software Proteus versión 6.0 en adelante.

4. CONCLUSIONES

- El compilador PCW es una herramienta poderosa que permite programar MICROCONTROLADORES PIC reduciendo el tiempo de programación y complejidad en los algoritmos.
- El PCW es un compilador muy completo ya que permite desde el manejo de una LCD hasta un protocolo de comunicación.
- En las pruebas realizadas se obtuvieron excelentes resultados al comparar el desempeño del microcontrolador programado en ensamblador y el programado en lenguaje C mediante el PCW.

5. BIBLIOGRAFÍA

[1] José María Angulo Uscategui, Diseño Practico de Aplicaciones de Microcontroladores PIC16F877, Mcgraw-hill, Segunda Edición, España 2000

[2] Jose María Angulo Uscategui, Diseño Práctico de Aplicaciones Primera Parte PIC16F84 Lenguaje Pbasic y Ensamblador, Tercera Edición, Mcgraw-hill, España 2003

[3] Andrés Cánova López, Manual de Usuario del Compilador PCW de CCS, Microchip, España 2000

[4] www.microchip.com